디자인 패턴

퍼사드(Facade)

복잡한 서브 클래스들의 공통적인 기능을 정의하는 상위 수준의 인터페이스를 제공하는 패턴이다.

퍼사드란 프랑스어로 건물의 외관이라는 뜻으로 건물의 외벽에서 보면 안의 구조는 보이지 않는다.

즉 많은 서브시스템(내부 구조)을 거대한 클래스(외벽)로 만들어 감싸서 편리한 인터페이스를 제공해 주는 디자인 패턴이다.

퍼사드 패턴을 통해 서브 시스템(SubSystem)들 간의 종속성을 줄여줄 수 있으며, 퍼사드 객체를 사용하는 곳(Client)에서는 여러 서브 클래스들을 호출할 필요 없이 편리하게 사용할 수 있다.

> 💡 예시) 전자레인지

[가정]

전자레인지를 통해서 음식을 데워 먹을려고 한다.

일반적으로 우리는 타이머를 설정하고 시작 버튼만 누르면 데운 음식을 먹을 수 있다.

만약 음식을 데우기 위해서

1. 타이머의 전원을 넣고,
2. 실행 신호를 보내고,
3. 마이크로파를 직접 작동시켜서

음식을 데워야한다면 프라이팬에 음식을 데우는것이 더 편할 것이다.

그래서 제조사는 단순한 버튼 동작을 통해서 편리한 인터페이스를 제공한다.

[전자레인지 구성]

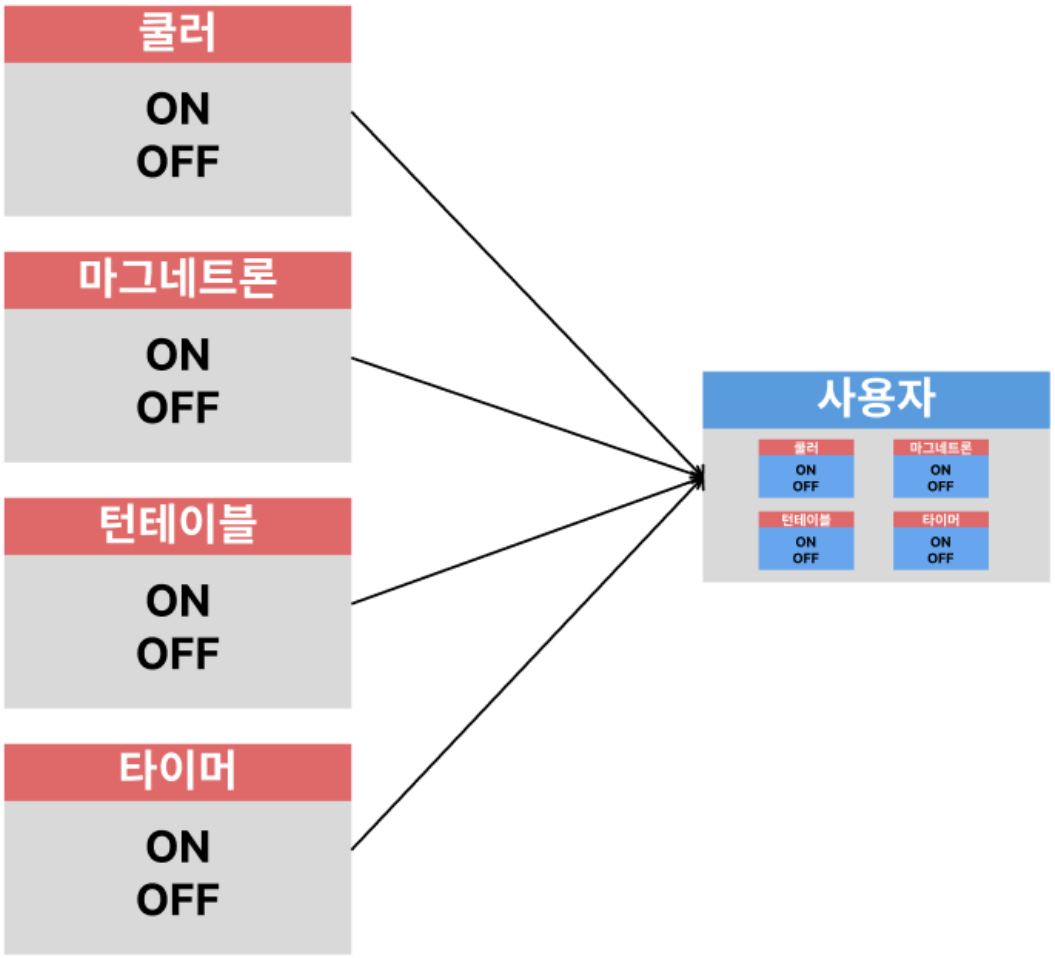
쿨러	전자레인지 열을 식힘
마크네트론	마이크로파 발생
턴테이블	조리할 음식 회전

타이머	시간이 되면 전원을 끔

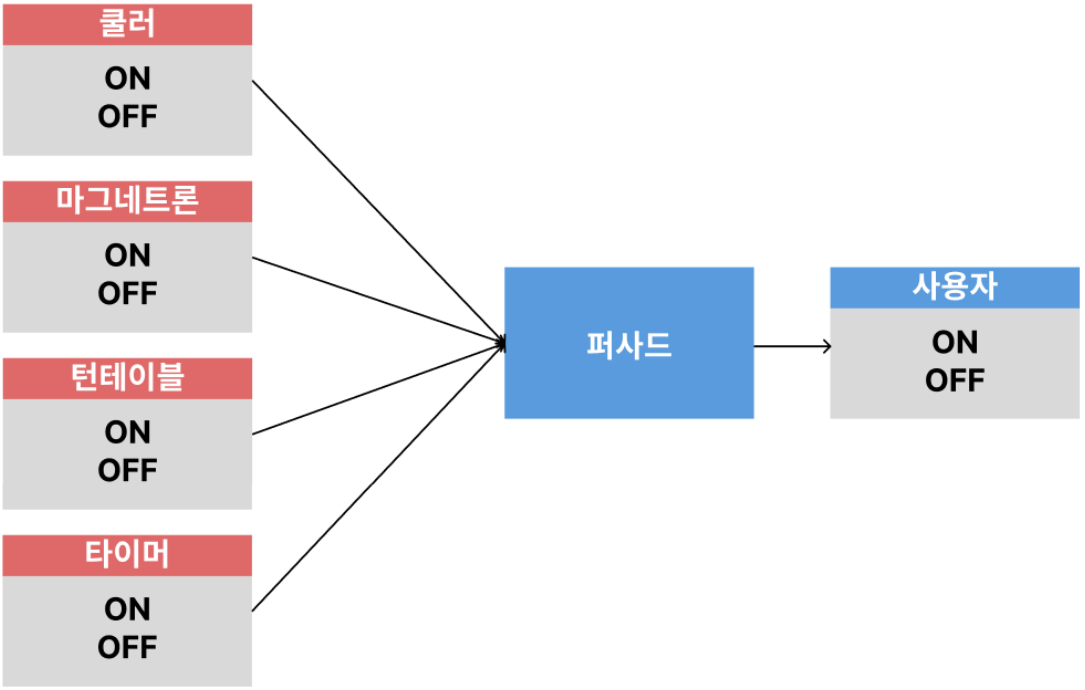
만약 퍼사드 패턴을 사용하지 않는다면 사용자는 직접 전원 스위치를 on시켜야한다.

> 쿨러를 작동시키고 → 마크네트론을 작동시키고→ 턴테이블을 돌린다음 → 타이머를 원하는 시간 만큼 작동시켜야 한다

그리고 정지를 시키려면 반대로 모든 동작을 직접 전원을 꺼야한다.



퍼사드를 사용하면 on/off버튼만으로 전자레인지를 작동 시킬 수 있다



플라이웨이트(Flyweight)

재사용이 가능한 객체 인스턴스를 공유시켜 메모리 사용량을 최소화하는 구조 패턴이다.

쉽게 말해 캐시 개념을 코드로 패턴화 한 것으로 말할 수 있다.

자주 변하는 속성과 자주 변하지 않는 속성을 분리하고 변하지 않는 속성을 캐시하여 재사용해 메모리 사용을 줄이는 방식이다.

동일하거나 유사한 객체들 사이에 가능한 많은 데이터를 서로 공유하도록 하여 최적화를 노리는 **경량패턴**이라고 불린다

플라이웨이트 패턴에서 중요한것은 intrinsic와 extrinsic 상태이다.

intrinsic은 '고유한, 본질적인' 이라는 뜻으로 인스턴스가 어떠한 사황에도 변하지 않는 정보를 말한다. 즉, 값이 고정되어있어 언제 어디서 공유해서 문제가 없다.

extrinsic은 '외적인, 비본질적인'이라는 의미로 인스턴스의 두는 장소나 상황에 따라 변화하는 정보를 말한다. 즉 값이 언제 어디서 변할지 모르기 때문에 캐시해서 공유 할 수는 없다.

💡 예시) 폭탄

예를 들어 폭탄 피하기 게임을 만든다고 해보자

생성되어 떨어지는 모든 폭탄은 하나의 객체 일텐데 하나하나 new를 통해서 인스턴스화 하면 폭탄이 생성될 때 마다 메모리를 차지하게 되어 엄청난 메모리를 차지하게 될 것이다.

여기서 생각해볼것이 intrinsic와 extrinsic 상태이다

폭탄피하기 게임이 가지고 있는 정보는 다음과 같다

- 폭탄의 모양
- 폭탄의 색깔
- 폭탄의 좌표(x,y)

폭탄의 모양이나 색깔은 변하지 않으니 공유가 가능하다

하지만 폭탄의 좌표는 항상 일정하지 않고 모두 다르기 때문에 공유를 할 수 없다.

intrinsic한 객체	폭탄의 모양, 폭탄의 색깔
extrinsic한 객체	폭탄의 좌표

이처럼 폭탄을 플라이웨이트로 처리함으로써 폭탄 인스턴스를 하나만 만들고 공유하고 이를 가져와 화면에 뿌리면 메모리 중복을 막을 수있고 인스턴스의 개수를 줄여서 메모리르 절약할 수 있다.